Distributed processing in virtual simulation using Cloud Computing Environment

Jarosław Koszela[1], and *Maciej* Szymczyk[1,*]

[1]Military University of Technology, Faculty of Cybernetics, gen. Witolda Urbanowicza 2, 01-476 Warsaw, Poland

Abstract. Today's hardware has computing power allowing to conduct virtual simulation. However, even the most powerful machine may not be sufficient in case of using models characterized by high precision and resolution. Switching into constructive simulation causes the loss of details in the simulation. Nonetheless, it is possible to use the distributed virtual simulation in the cloud-computing environment. The aim of this paper is to propose a model that enables the scaling of the virtual simulation. The aspects on which the ability to disperse calculations depends were presented. A commercial SpatialOS solution was presented and performance tests were carried out. The use of distributed virtual simulation allows the use of more extensive and detailed simulation models using thin clients. In addition, the presented model of the simulation cloud can be the basis of the "Simulation-as-a-Service" cloud computing product.

1 Introduction

Science and technology advance more and more rapidly. Moore's law is a good example of this phenomenon. Additionally, our opportunities and requirements increase as well as the needs of everyday, social and economic life.

Technological development of computing units is specific. CPU clock rate on a commercial market has not exceeded 4-5 GHz for a long time. It is due to the properties of silicon used to build processors. Therefore, acceleration is achieved by multiplying cores and system organization so as to enable concurrency and distribution [1].

Three types of simulation may be distinguished. Actual simulation makes people use physically existing systems according to the prepared scenario. Virtual simulation requires interaction in the computer-simulated environment. It may also require decision-making, motricity and communication skills. Constructive simulation is a computer program, whose operations are based on aggregated simulation objects. They represent actual situations in the context of the simulation scenario [2].

Virtual simulation has higher resolution and conformity with the real world than the constructive simulation. Unfortunately, in case of large, detailed and/or dynamic simulations, it may turn out that the computing power of a single device is insufficient. One of the possibilities is to switch into the constructive simulation. Unfortunately, many objects aggregated into one may cause the loss of details and necessity to calibrate its features. It is possible to distribute the virtual simulation into many computers/computing nodes. Therefore, the computing power of one device is not a limiting factor.

The article consists of three sections. The first one deals with models, mechanisms and issues currently used in computer games and virtual simulation. In the second one, the possible directions of development of the virtual simulation are presented and described. The third section is dedicated to performance tests using the SpatialOS platform.

2 Current approach

2.1 Game loop

An engine of each computer game and virtual simulation is based on a certain type of a loop. During one loop, input devices are handled, whereas the calculated physics, logic, artificial intelligence and rendered graphics displayed on screen. The above-mentioned steps in the classic approach are sequential. Only after the loop stops, the other one may start.

* Corresponding author: maciej.szymczyk@wat.edu.pl

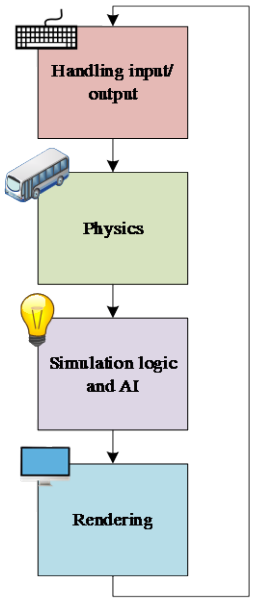

Fig. 1. Sequential game engine model.

To maintain the game flow, screen frames must be displayed at a certain frequency. To obtain 50 frames per second, all computing related to a single loop must last maximum 20 milliseconds. The tolerable lower limit for games is 16 frames [3]. Otherwise, the game shall not be steady. Table 1 shows this phenomenon by comparing the number of objects with frames rendered per second. The data were obtained in the "Pirates" project on SpatialOS executed using virtual machines, with the following parameters: 2 cores of i7 4790k 4 Ghz processor and 4 GB RAM memory.

Table 1. Comparison of the number of objects and frames per second for a single machine.

Number of objects	Frame time (ms)	FPS
80	20	50
130	20	50
280	27	37
400	50	20
530	357	2.8
1030	769	1.3

When introducing the architecture of multicore processors, it turned out that the sequential game engine proved inefficient. Therefore, a perfect solution is asynchronous loop, where tasks do not have to wait for the results of other tasks. Instead, the most recently calculated result taken from a shared resource is considered. Unfortunately, such model may be too difficult to be implemented in practice. Some tasks must be performed sequentially.

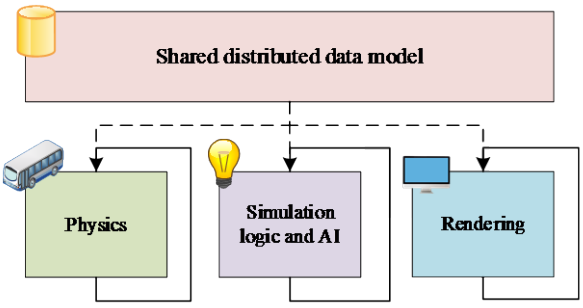

Fig. 2. Asymmetric game engine model

To allow simultaneous execution of tasks in the game engine loop, it is possible to divide the data in the parallel sections of the application. Instead of using one main loop, separate threads may process the data sets. The tasks that must be executed sequentially as well as the necessity to exchange messages between the objects from different sections constitute limiting factors in this model [4]. Events should be able to spread. An example of a simulation in which this phenomenon may occur is the simulation of contamination threats [5].

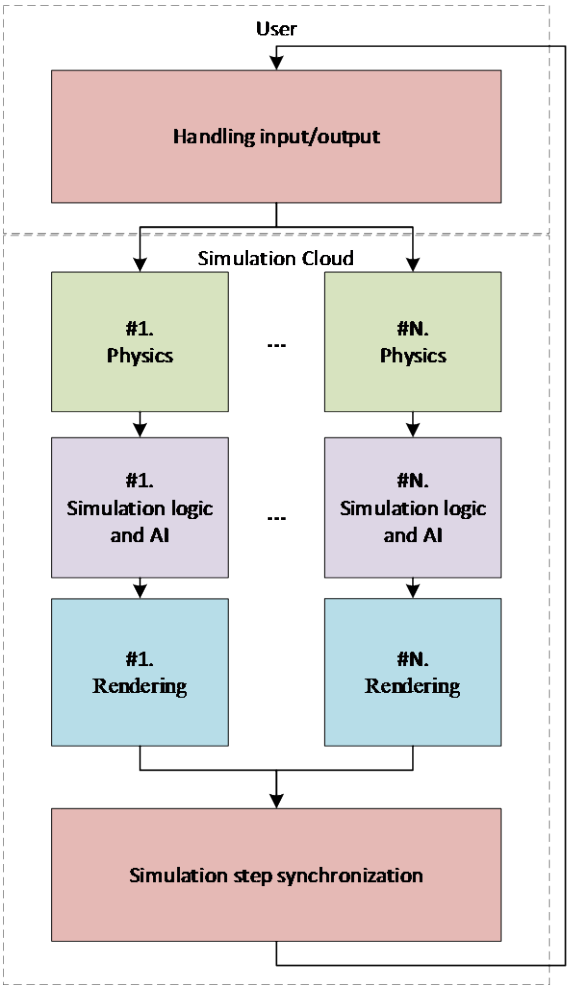

Fig. 3. Asymmetric game engine model with data separation.

2.2 Distributed Simulation

In case of computer games, peer-to-peer and client-server are the most often used types of multiplayer game architecture.

The main advantage of peer-to-peer is the absence of a central server. The cost of creating and maintaining such network is low. Unfortunately, the solution is not scalable. The speed of connections limits the capacity and the implementation is complex. Slow connection of one of the clients may cause delays in the gameplay of others. Not only some problems with synchronization occur, but also it is necessary to keep a fixed number of frames for all clients. When using P2P, fraud prevention is difficult.

The client-server architecture requires one specific machine, which shall be used as a server and make the service available to clients. The users use client applications that communicate with the server. The applications exchange messages and receive a list of changes that need to be made in the local virtual world [6].

The server may act as an intermediary in the process of exchanging messages between clients or actively participate in the gameplay. In such case, it is the server that decides about the occurring events (e.g. which player shot the first). The server also protects against frauds.

Some delays caused by the network infrastructure may make it difficult to find pleasure in the gameplay and ensure its realism. As a result, the differences in the object status on the game server and client's devices may occur. A possible solution to this problem is to use the prediction mechanism, which estimates future properties of the object based on current data (e.g. position, speed, direction).

The architecture of traditional network games assumed the use of one server in the client-server architecture. In case of MMO RPG games, i.e. very complex and developed games for multiple players, one server is insufficient to handle such a large number of connections and data. Therefore, many servers are necessary. Such servers may be independent (statuses on game progress are stored on each server separately, without synchronization) or have a common database. In the first case, the user may choose the server to log in. In the second case, there are machines counteracting movement on servers and assigning users to the least loaded machine [7].

In the context of simulators, the following two types of architecture are most often used: Distributed Interactive Simulation (DIS) and High Level Architecture (HLA).

DIS is the information exchange standard between simulators. It does not have any central management server. The simulators may join and leave the simulation at any moment.

The simulation status is saved in a message called the Protocol Data Unit (PDU) and exchanged between simulators through available transport layer protocol, using multicast or broadcast method. The current version of DIS 7 defines 72 various types of PDU [8].

High Level Architecture is the architecture designed for computer systems (in particular, simulation systems). The idea of this solution is based on making the communication independent of platforms, where the systems are installed.

The simulator in compliance with the HLA is called a federate. The systems are connected with the Runtime Infrastructure (RTI) and create together the so-called Federations. The system elements share data on objects, which are in the Federation Object Model (FOM). Additionally some events (interactions) with parameters are sent between the simulators [9].

Unfortunately, the application of the aforementioned types of architecture for many simulators has one main drawback. From a logical point of view, the virtual world is simulated on every federate. The main purpose of the DIS and HLA architecture is to synchronize different types of simulators. It does not allow to distribute the virtual world between different simulators.

3 Cloud simulation

3.1 Cloud GPU rendering

The use of computing cloud solutions in the virtual simulation suggests the use of thin clients. Currently, fat clients with quick processors and graphic cards are used.

It would be possible to use thin clients only if the graphics from each client were processed by the server. In the computer game environment, this phenomenon is called cloud gaming. Among others, NVIDIA under business name NVIDIA GRID, offers such service.

The solution allows to send input data from the client to the server. The client receives video stream, which is decoded and displayed on the screen. Therefore, the game or simulation is independent of the platform that the client has to have [10].

3.2 Computing nodes

One of the features that the simulation cloud needs to have is an asynchronous engine, which allows to scale in a horizontal direction. In such way, it is possible to initialize computing nodes, as the need arises. It is important for each node to be responsible for various simulation objects and to allow interaction between objects operated by other nodes.

The objects may be assigned to computing nodes on a territorial basis. When the object is within the area operated by a given node, then it is operated by such node. The area sizes may differ. For example, cities are more densely filled with simulation objects than forests.

The nodes are initialized and assigned both statically and dynamically. During simulation and analysis, it is possible to prepare satisfactory distribution of the assigned computing nodes. The operation should be repeated for each scenario. The dynamic assignment means continuous analysis of the load of computing nodes, initiation of new nodes and release of such nodes.

Such mechanism allows to scale resources depending on the simulation complexity [11, 12].

An advantage of the dynamic distribution is flexibility and economy of the cloud resources if their use is indispensable. On the other hand, some load may occur due to testing and analysis of the current and predictable load of the nodes. Another advantage of the dynamic distribution of computing nodes is greater resistance to failure. In case of unexpected loss of functionality of the node, a new node is initiated and the simulation continues. The use of the simulation cloud as a service allows steady initialization of new instances and control of the simulation costs [13].

Apart from the standard computing units, it is possible to use dedicated and specialist units. A simple game engine model supports physics, logic and artificial intelligence. The dedicated nodes may use specialist graphic cards, processors or components supporting the artificial intelligence and machine learning [14].

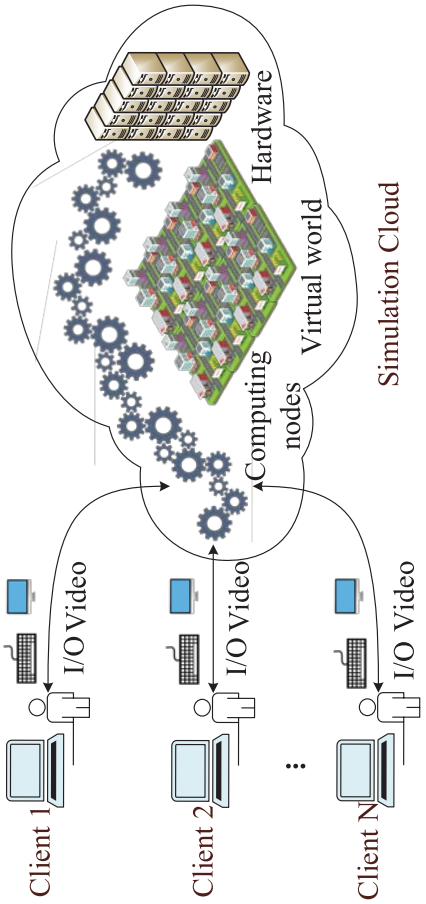

Fig. 4. Simulation Cloud Model

3.2 SpatialOS

The SpatialOS platform by Improbable is one of the commercial products using the distributed processing in the computing cloud for virtual simulations. The solution is based on the Google Cloud Platform.

The virtual simulation in SpatialOS is operated by the computing nodes known as Workers. In case of increased need for the simulation computing capacity, new instances of the computing nodes are initiated. Two types of workers are distinguished: managed and external. The managed workers are those, whose lifecycle is managed by SpatialOS. The external workers are usually clients. SpatialOS does not determine when the external worker connects or disconnects.

All types of workers are connected to the SpatialOS platform. If an event occurs in one of the nodes (client or worker), it is sent to SpatialOS, which informs about this other nodes. Only workers may introduce changes in the virtual world. In case of the clients, the events are handled only to display 3D models, textures and animations.

Every worker is responsible for only a part of the virtual world. The managed workers deal with computing of the part of the world assigned thereto by the SpatialOS platform. The external workers are only aware of the part of the world, which surrounds the simulation object (e.g. a character), with which they are connected.

As part of the project, it is possible to configure the method of organization of the computing nodes. In case of the static distribution, node coordinates and number are established. The dynamic distribution requires determination of the maximum number of nodes as well as auto-scaling on the basis of the maximum number of the operated objects [15].

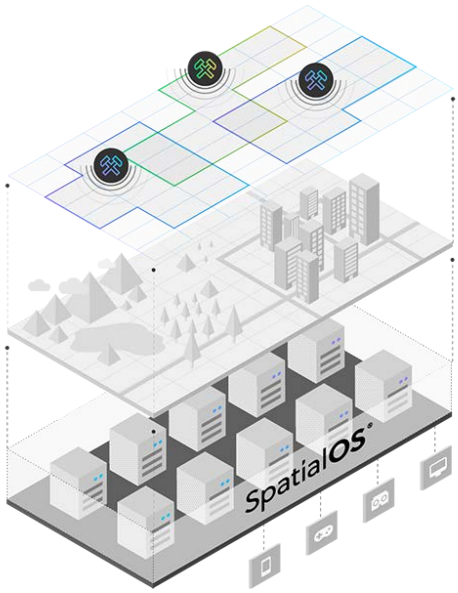

Fig. 4. SpatialOS model

4 Efficiency tests

The purpose of the aforesaid tests was to discern the capacity differences between the simulations based on 1, 2 or 4 computing nodes. The simulation was based on the SpatialOS platform, using the "Pirates" project. The static method of node distribution was used in the scenario. 4 positions of the nodes on coordinates were determined: <-250,-250>, <-250,250>, <250,-250> and <250,250>. The objects were placed randomly, at a

distance of 500 units from position <0,0>. Figure 5 shows location of nodes and objects.

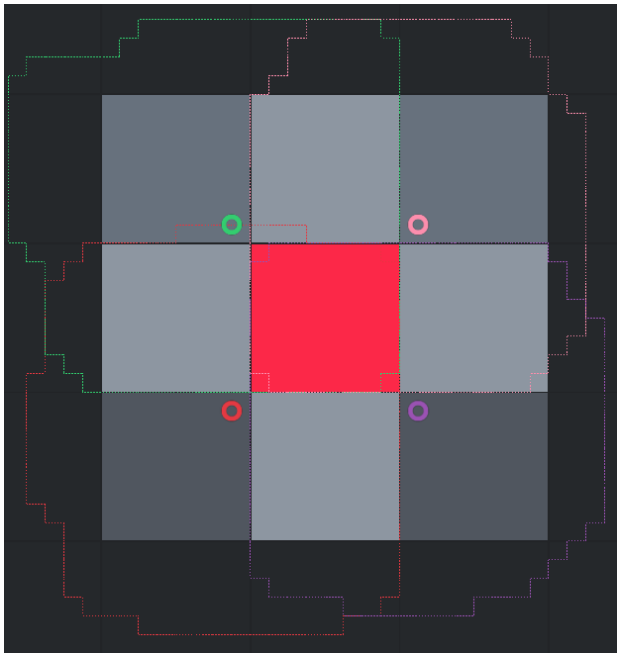

Fig. 5. Worker nodes placement and simulation objects density

The test was performed in the configuration of 1, 2, 4 nodes and 280, 530, 780, 1030, 1280 and 1530 objects, out of which 30 were static elements of the surrounding. The rest were pirate ships moving towards a randomly chosen direction. The test results are in table no. 2.

Table 2. Comparison of the number of objects and frames per second for many computing nodes.

Number of simulation objects	Number of worker nodes					
	1		**2**		**4**	
	FPS	Objs/ node	FPS	Objs/ node	FPS	Objs/ node
280	44	280	59.8	208	60	147
530	29	530	43.5	383	56.4	273
780	3.3	780	33	574	48.1	402
1030	2.2	1030	23.6	744	28.2	540
1280	2.1	1280	3.3	911	6.5	653
1530	1.5	1530	2	1086	2.4	773

The test results show that the increase in the number of frames per second is not linear in comparison with the increase in the number of computing nodes. One node is not enough for 780 objects. Figure 5 shows a chart bar presenting data from table 2. The application of 4 nodes allowed smooth simulation at the level of around 30 fps with 1030 objects.

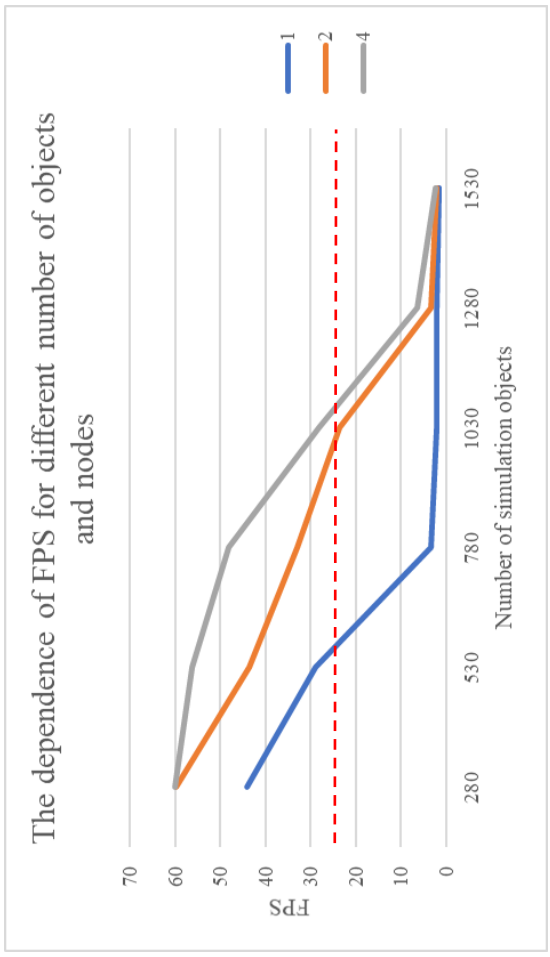

Fig. 5. The dependence of FPS for different number of objects and nodes line chart

The greatest increase in FPS is noticeable when switching from 1 node to 2 nodes. The lowest increase in FPS is noticeable when switching from 2 node to 4 nodes. The phenomenon is illustrated in figure 7. Therefore, it is evident that the use of a larger number of nodes makes it necessary to use additional resources to operate and synchronize such nodes.

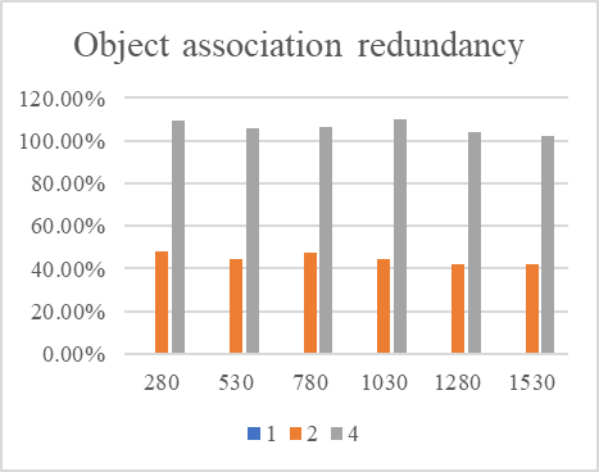

Fig. 6. Object association redundancy bar chart

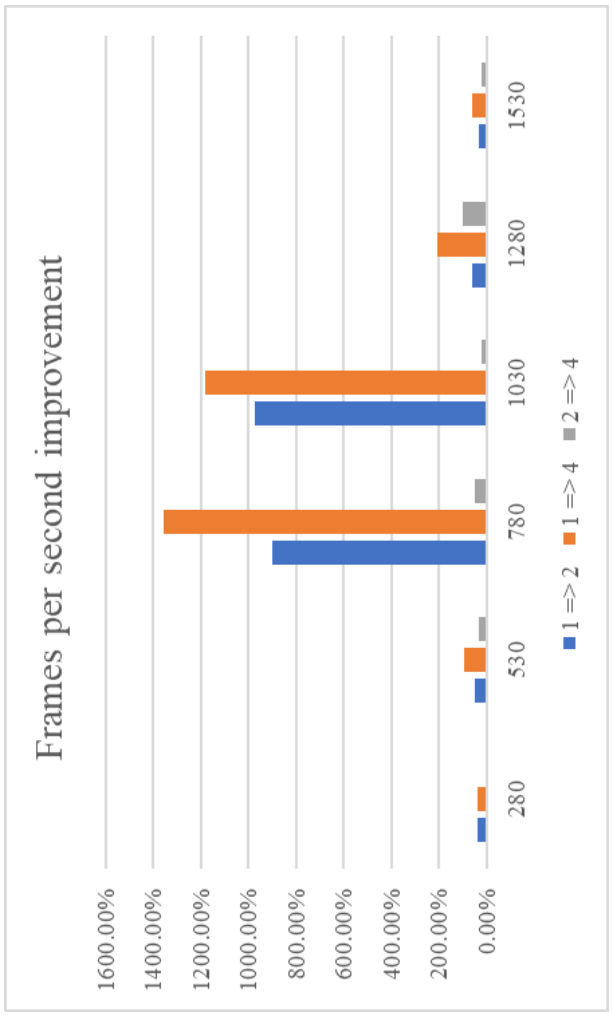

Fig. 7. Frames per second improvement bar chart

Low increase in capacity probably results from high redundancy of objects assigned to nodes, which is shown in figure 6. In case of 4 computing nodes, the sum total of all objects in nodes is twice as high as the number of all objects in the simulation. It is caused by overlapping of responsibility areas of nodes and high density of distribution of objects between all nodes. In case of even distribution of simulation objects, the redundancy shall be lower, and the FPS increase higher.

5 Conclusions

The article discussed the concept of using distributed virtual simulation. The authors showed typical problems currently faced by the virtual simulation and presented solutions using the potential of the computing cloud for the distributed virtual simulation.

The application of many nodes requires a lot of computing power to manage and exchange messages between them. According to Amdahl's law, it is important to limit the execution of the sequential code.

The existing solutions may be extended with the rendered graphics in cloud (e.g. NVIDIA GRID). The article outlined the concept and simple capacity tests of the SpatialOS platform. The use of the distributed environment allowed to double the size of the simulation with 4 computing nodes. In light of the above, the increase of performance is not linear. The more nodes, the smaller increase of performance.

The scenario, in which the objects did not change their location in a short period of time, was used for the purpose of testing. In practice, the changing virtual environment may cause differences in load of the computing nodes. Further research shall be devoted to methods and techniques of assignment of tasks (simulation objects) to the computing nodes.

The article presents possible directions of development of a distributed virtual simulation. The efficiency gains for 2 and 4 computing nodes on the SpatialOS platform were tested and discussed. Future work will focus on developing an efficient model of management and communication of computing nodes as one of the simulation cloud modules. Development in the field of distributed virtual simulation may result in the departure from the use of constructive simulation, i.e. the need to aggregate simulation objects.

References

1. P. P. Mattsson, Why Haven't CPU Clock Speeds Increased in the Last Few Years? https://www.comsol.com/blogs/havent-cpu-clock-speeds-increased-last-years/ (accedes on 12.03.2018)
2. R. M. Weatherly, A. L. Wilson, B. S. Canova, E. H. Page, A. A. Zabek, M. C. Fisher, Advanced Distributed Simulation though the Aggregate Level Simulation Protocol, *Proceedings of the 29th Hawaii International Conference on Systems Sciences*, 407, (1996)
3. M. Joselli, M. Zamith, L. Valente, B. Feijó, F.R. Leta, E. Clua, A Distributed Architecture for Simulation Environments Based on Game Engine Systems, *Augmented Vision and Reality*, **4**, 14, (2014)
4. Multithreaded Game Engine Architectures, www.gamasutra.com/view/feature/130247/multithreaded_game_engine_.php?print=1 (accessed on 12.03.2018)
5. Z. Tarapata, R. Antkiewicz, M. Chmielewski, M. Dyk, R. Kasprzyk, W. Kulas, A. Najgebauer, D. Pierzchała, J. Rulka, A Computer System or CBRN Contamination Threats Analysis Support, Prediction Their Effects and Alarming the Population: Polish Case Study, *21st International Conference on Circuits, Systems, Communication and Computers*, **125**, (2017)
6. R. Cohen, A. Ejlersen, R. Kristensen, *Distributed Game Engine for Massively Multiplayer Online Shooting Games, 10-12,* (2009)
7. K. Emilsson, Infinite Space: An Argument or Single-Sharded Architecture in MMOs https://www.gamasutra.com/view/feature/132563/in

finite_space_an_argument_for_.php?print=1 (accedes on 12.03.2018)

8. 1278.1-2012 – IEEE Standard for Distributed Interactive Simulation—Application Protocols, *IEEE,* (2012)
9. 1516-2010 – IEEE Standard for Modelling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules, *IEEE,* (2010)
10. NVIDIA GRID: Graphics accelerated VDI with the visual performance of a workstation http://www.nvidia.com/content/grid/vdi-whitepaper.pdf (accedes on 12.03.2018)
11. O. Severiukhina, P. A. Smirnov, K. Bochenina, D. Nasonov, N. Butakov, Adaptive load balancing of distributed multi-agent simulations on heterogenous computational infrastructures, *6th International Young Scientists Conference in HPC and Simulation*, (2017)
12. Q. Bragard, A. Ventresque, L. Murphy, Self-Balancing Decentralized DisAQtributed Platform for Urban Traffic Simulation, *IEEE Transactions on Intelligent Transportation Systems,* **18,** 1190-1196, (2017)
13. W. Xiong, W. Tsai, HLA-Based SaaS-Oriented Simulation Frameworks, *IEEE 8th International Symposium on Service Oriented System Engineering*, 1-7, (2014)
14. Google's dedicated TensorFlow processor, or TPU, crushes Intel, Nvidia in inference workloads www.extremetech.com/computing/247199-googles-dedicated-tensorflow-processor-tpu-makes-hash-intel-nvidia-inference-workloads (accedes on 12.03.2018)
15. SpatialOS SDK Documentation, docs.improbable.io (accessed on 12.03.2018)